author	Hawkins Jiawei <yin31149@gmail.com>	2022-10-18 10:18:51 +0800
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-11-10 18:15:30 +0100
commit	aa16cac06b752e5f609c106735bd7838f444784c (patch)	
tree	8c1807ac5eca397250932115794e6f5e5218b666	
parent	a3a7b2ac64de232edb67279e804932cb42f0b52a (diff)	
download	linux-aa16cac06b752e5f609c106735bd7838f444784c.tar.gz	

Bluetooth: L2CAP: Fix memory leak in vhci_write

```
[ Upstream commit 7c9524d929648935bac2bbb4c20437df8f9c3f42 ]

Syzkaller reports a memory leak as follows:
===================================
BUG: memory leak
unreferenced object 0xffff88810d81ac00 (size 240):
  [...]
  hex dump (first 32 bytes):
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  backtrace:
    [<ffffffff838733d9>] __alloc_skb+0x1f9/0x270 net/core/skbuff.c:418
    [<ffffffff833f742f>] alloc_skb include/linux/skbuff.h:1257 [inline]
    [<ffffffff833f742f>] bt_skb_alloc include/net/bluetooth/bluetooth.h:469 [inline]
    [<ffffffff833f742f>] vhci_get_user drivers/bluetooth/hci_vhci.c:391 [inline]
    [<ffffffff833f742f>] vhci_write+0x5f/0x230 drivers/bluetooth/hci_vhci.c:511
    [<ffffffff815e398d>] call_write_iter include/linux/fs.h:2192 [inline]
    [<ffffffff815e398d>] new_sync_write fs/read_write.c:491 [inline]
    [<ffffffff815e398d>] vfs_write+0x42d/0x540 fs/read_write.c:578
    [<ffffffff815e3cdd>] ksys_write+0x9d/0x160 fs/read_write.c:631
    [<ffffffff845e0645>] do_syscall_x64 arch/x86/entry/common.c:50 [inline]
    [<ffffffff845e0645>] do_syscall_64+0x35/0xb0 arch/x86/entry/common.c:80
    [<ffffffff84600087>] entry_SYSCALL_64_after_hwframe+0x63/0xcd
===================================

HCI core will uses hci_rx_work() to process frame, which is queued to
the hdev->rx_q tail in hci_recv_frame() by HCI driver.

Yet the problem is that, HCI core may not free the skb after handling
ACL data packets. To be more specific, when start fragment does not
contain the L2CAP length, HCI core just copies skb into conn->rx_skb and
finishes frame process in l2cap_recv_acldata(), without freeing the skb,
which triggers the above memory leak.

This patch solves it by releasing the relative skb, after processing
the above case in l2cap_recv_acldata().

Fixes: 4d7ea8ee90e4 ("Bluetooth: L2CAP: Fix handling fragmented length")
Link: https://lore.kernel.org/all/0000000000000d0b1905e6aaef64@google.com/
Reported-and-tested-by: syzbot+8f819e36e01022991cfa@syzkaller.appspotmail.com
Signed-off-by: Hawkins Jiawei <yin31149@gmail.com>
Signed-off-by: Luiz Augusto von Dentz <luiz.von.dentz@intel.com>
Signed-off-by: Sasha Levin <sashal@kernel.org>
```

Diffstat

-rw-r--r-- net/bluetooth/l2cap_core.c 7

1 files changed, 3 insertions, 4 deletions

```
diff --git a/net/bluetooth/l2cap_core.c b/net/bluetooth/l2cap_core.c
index a18a1c608ed127..4802583f107192 100644
--- a/net/bluetooth/l2cap_core.c
+++ b/net/bluetooth/l2cap_core.c
@@ -8461,9 +8461,8 @@ void l2cap_recv_acldata(struct hci_conn *hcon, struct sk_buff *skb, u16 flags)
                 * expected length.
                 */
                if (skb->len < L2CAP_LEN_SIZE) {
-                       if (l2cap_recv_frag(conn, skb, conn->mtu) < 0)
-                               goto drop;
-                       return;
+                       l2cap_recv_frag(conn, skb, conn->mtu);
+                       break;
                }

                len = get_unaligned_le16(skb->data) + L2CAP_HDR_SIZE;
@@ -8507,7 +8506,7 @@ void l2cap_recv_acldata(struct hci_conn *hcon, struct sk_buff *skb, u16 flags)
                        /* Header still could not be read just continue */
                        if (conn->rx_skb->len < L2CAP_LEN_SIZE)
                                return;
+                               break;
                }

                if (skb->len > conn->rx_len) {
```